MOUGAMADOUBOUGARY Mohamed
MARSSO Denn

COMPTE RENDU PROJET C++
World Cup Ultimate Team

DESCRIPTION DE L'APPLICATION :

Pour notre projet, nous avons décidé d'effectuer une draft FIFA qui est jouable en mode attaque/défense, en s'inspirant du jeu FIFA produit par EA SPORTS.

FIFA est un jeu de foot mondialement connu qui possède différents modes de jeu, l'un de ces modes est la draft. Elle consiste à créer une équipe en choisissant les joueurs qui sont générés aléatoirement.

Voici à quoi ressemble ce mode de jeu. On a donc 11 cartes qui apparaissent à l'écran, en appuyant sur les cartes vides, on a un joueur :

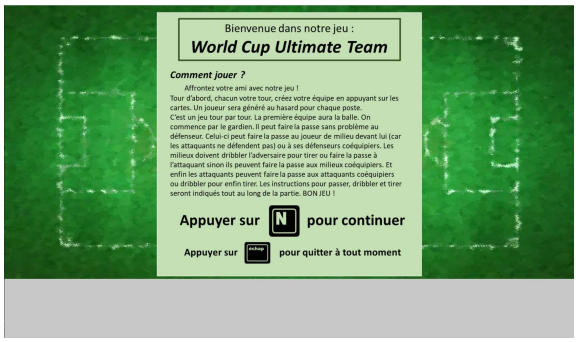
Exemple de Draft sur FIFA23

Le but est d'avoir les meilleurs joueurs possible et donc la meilleure note.

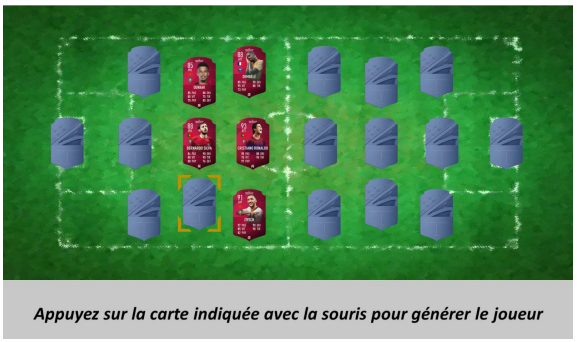
Notre jeu ressemblera à ça avec certains ajouts. Nous aurons donc un premier écran qui sera le menu dans lequel les règles seront expliquées, et les commandes ,pour changer d'écran et pour quitter, seront indiquées (figure *Ecran 1*). Puis l'écran suivant affiche les cartes vides pour générer les joueurs. Nous aurons 20 cartes, 10 pour chaque équipe donc 10 sur chaque côté du terrain et lorsque le joueur cliquera dessus son joueur apparaîtra (figure *Ecran 2*).
Après avoir généré les 20 joueurs, en cliquant sur un bouton le jeu passera à l'écran suivant dans lequel ces cartes seront remplacées par des cercles, seront mis à leur poste respectif et les instructions seront affichées pour jouer (figure *Ecran 3*).

C'est un jeu de tour à tour. La balle sera donnée, au début, au gardien de l'équipe 1 (donc à gauche) et le 1er joueur pourra attaquer, il pourra dribbler, passer ou tirer en

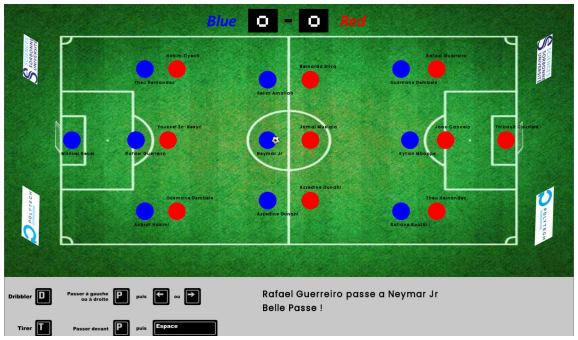
fonction du joueur qui à la balle et de sa position. Si il marque ou si il se fait prendre la balle (dribble qui échoue, ou tir loupé) la balle est donnée au gardien adverse et ça sera au second joueur d'attaquer. Le premier arrivé à 5 buts gagne le match, vous pouvez recommencer en appuyant sur la touche R (figure *Ecran 4*).



Ecran 1



Ecran 2



Ecran 3

Ecran 4

DIAGRAMME DE CLASSE :

Avant d'effectuer un diagramme complet du jeu, nous avons d'abord fait les différentes classes pour les joueurs et le ballon :

- Joueur : super-classe abstraite (car elle a 2 fonctions virtuelles poste() et getVitesse())
- JoueurDeChamp, Gardien : sous-classe héritant de la classe Joueur
- Attaquant, Defenseur : sous-classe héritant de la classe JoueurDeChamp
- Milieu : sous-classe héritant des classes Attaquant et Defenseur.
- Ballon

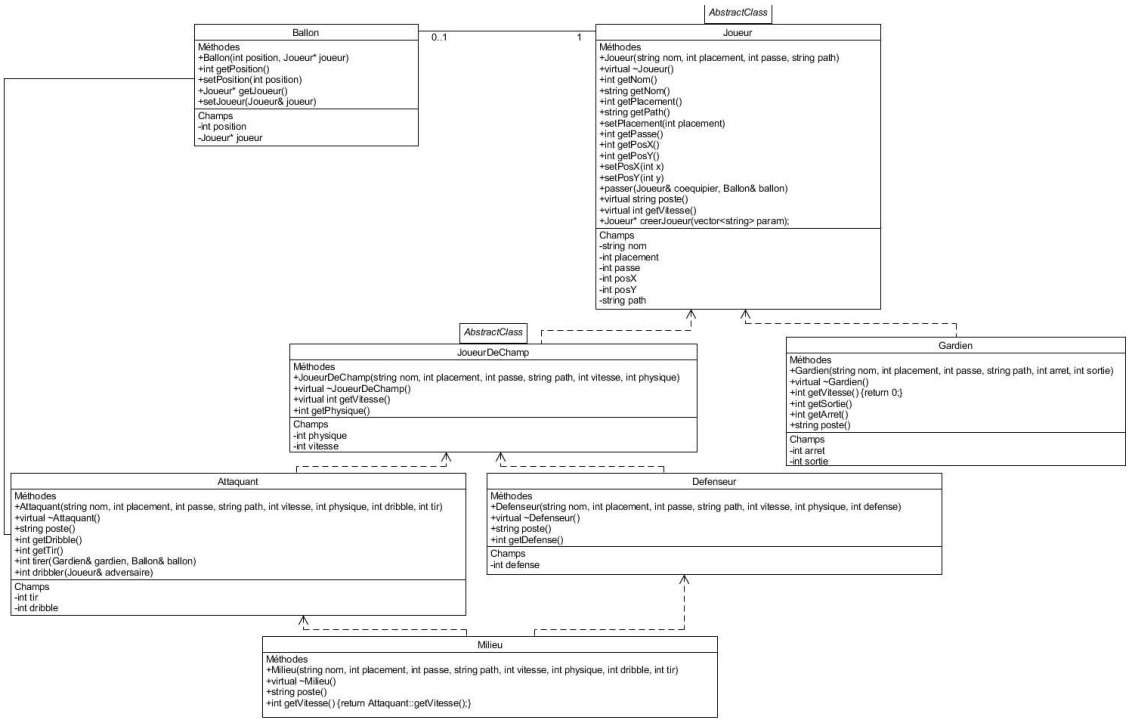


Diagramme de classe initial

On a ensuite rajouter la classe permettant de modéliser la base de données du jeu (listes d'attaquants, de défenseurs, de milieux, de gardiens, listes des joueurs des équipes… etc) puis on a les classes Application et Input qui permettent de gérer la partie graphique et les entrées avec les touches (input).

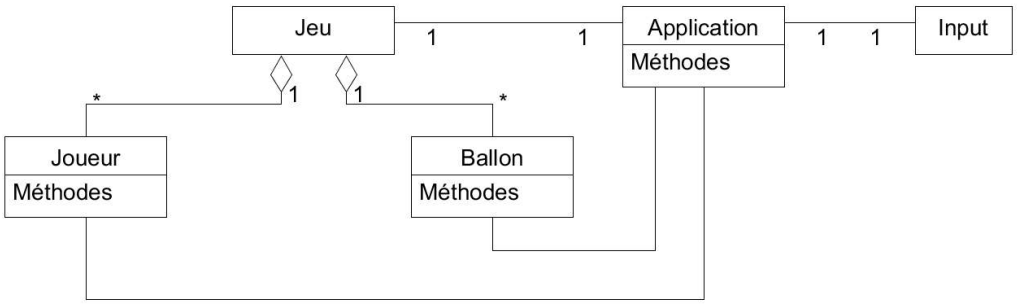


Diagramme de classe final

On a donc 10 classes au total.

EXPLICATIONS DU CODE (*en utilisant SFML*):

Dans le main, on instancie une Application et lance la méthode run(), ce qui fera tourner en boucle le jeu avec la partie graphique avec une boucle while(). En instanciant Application, on instancie Jeu. Grâce à la classe Jeu, on initialisera les joueurs en lisant la base de données : ce qui créera une liste d'attaquants, une liste de milieux, une liste de défenseurs et une liste de gardiens.

Pour la partie graphique, on crée une fenêtre, on charge les éléments à afficher et les textes. Grâce au switch-case pour les écrans, on reste dans l'écran menu tant qu'on appuie pas sur la touche N. Pour cela, on appelle la fonction CheckBtn() qui vérifie si un bouton est appuyé et appelle d'autres fonctions selon le bouton appuyé.

Avant de passer au 2e écran, on appelle des fonctions de la classe Jeu pour initialiser tous les paramètres de jeu comme par exemple les listes des joueurs pour chaque poste.

Pour le 2e écran on affiche les 20 cartes avec une image d'une carte vide. Puis en faisant appel à la fonction CheckBtn() à chaque tour de boucle, on vérifie si un clic gauche de la souris a été fait sur une des cartes(en vérifiant si le curseur de la souris est à l'intérieur de l'image de la carte). Si on appuie sur une carte, on fait appel aux fonctions choixAttaquant(), choixMilieu(), choixDefenseur() ou choixGardien() de la classe Jeu qui génère au hasard un joueur d'un poste précis selon le placement de la carte. Donc on change l'image de la carte vide par la carte du joueur en changeant la "Texture". Pour avoir la carte du joueur, chaque joueur a en argument le chemin de l'image par exemple "cartes/mbappe.png" (on a l'image du Joueur Mbappe dans le dossier cartes). En appuyant sur les cartes, chaque joueur généré sera mis dans la liste de l'équipe (équipe 1 pour les 10 cartes à gauche et équipe 2 pour les 10 cartes à droite).

Après avoir généré les 20 cartes, on appuie sur N pour passer à l'écran suivant.
Et juste avant de passer au 3e écran, on initialise les placements et les positions *x* et *y* de chaque joueur grâce à la fonction initPlacement() de la classe Jeu. Les valeurs de *x* et *y* ont été calculées et corrigées par rapport à la taille de la fenêtre. Donc les positions *x* et *y* des cercles correspondent à celles des joueurs et pendant le match on lit les positions des joueurs pour mettre à jour les positions des cercles grâce à la fonction updatePositionCircle(). Pour le 3e écran, on affiche les 20 cercles qui représentent les joueurs et leur nom sera affiché. A l'exception des gardiens, chaque joueur est en face d'un adversaire : défenseur contre attaquant et milieu contre milieu (voir figure *Ecran 3*). On indique que le possesseur du ballon est le gardien de l'équipe 1. On fait donc appel à la fonction CheckAction() pour vérifier si les touches pour dribbler, tirer et passer ont été appuyées.

En suivant les instructions indiquées ci-dessous, le match peut commencer et la première équipe qui arrive à 5 buts remporte le match. On aura donc le 4e écran et vous pouvez appuyer sur la touche R pour revenir au 2e écran et donc recommencer la partie en régénérant une nouvelle équipe.

COMMENT FONCTIONNE LES FONCTIONS DRIBBLER, TIRER ET PASSER ?

Tout d'abord, on a des stats pour chaque joueur. Pour les méthodes dribbler et tirer, on compare les stats du joueur qui appelle la fonction (le possesseur de la balle) et de celui qu'il affronte (celui en entrée de la fonction), en fonction du résultat de la comparaison il aura plus ou moins de chance de marquer ou de dribbler. Si il a des meilleures stats il aura 7 chances / 10, si il a les mêmes il en aura 4 et si il a des plus faibles stats que l'adversaire il en aura 2. S'il échoue de marquer ou dribbler la fonction renvoie 0.
Pour passer on fixe seulement le pointeur de joueur de la classe ballon au joueur en entrée de la fonction passer.

Expliquons maintenant comment sont utilisées ces méthodes.

On a donc les deux équipes avec tous les joueurs et leur position correspondante, il faut maintenant les faire s'affronter. Au départ, on met le gardien comme pointeur de joueur dans le ballon, et tout le reste du code se fait avec des appels de la fonction CheckAction() (la touche D pour dribbler, la touche T pour tirer et pour passer il faut appuyer en premier sur P et ensuite appuyer sur les flèches pour faire la passe sur les côtés ou alors sur Espace pour faire la passe devant).
Avant d'expliquer l'action de ces touches dans le code, on va expliquer le principe de position.
Les joueurs ont tous une position qui a été fixée après la draft , pour les attaquants c'est entre 30 et 32 ou -30 et -32, les milieux : 20 et 22 ou 20 et -22 , les défenseurs 10 et 12 ou -10 et -12 et enfin les gardiens 1 ou -1. Ces positions vont nous permettre de différencier les joueurs qui sont pris et les joueurs qui sont libres, un joueur qui dribble un autre joueur voit sa position augmenter de 5 ou diminuer de 5 en fonction de l'équipe, on pourra donc déplacer les cercles des joueurs pour montrer qu'il a réussi le dribble. Donc si il a une position différente de celles dites au dessus, on sait qu'il peut tirer (ou faire la passe devant si c'est un milieu).
Cela nous permettra aussi de connaître l'équipe du joueur qui a la balle, on vérifiera juste si la position est négative ou positive avant chaque méthode pour la faire correspondre à la bonne équipe.

Dans cette fonction, en fonction de la touche du clavier que l'utilisateur touche, le possesseur du ballon va effectuer des actions. Si il clique sur T, donc le bouton tirer dans la fonction input, la méthode tirer sera lancée pour le possesseur du ballon

contre le gardien adverse si le possesseur du ballon peut tirer, on vérifie donc la position du joueur, et l'équipe à laquelle il appartient avant de tirer. Il faut que ça soit un milieu ou un attaquant qui a déjà éliminé son adversaire, donc on vérifie bien le poste du joueur qui tire puis on vérifie si sa position à été implémentée de 5 ou non (donc si il a dribblé l'adversaire). S'il marque le score s'implémente de 1 sinon le score reste pareil, dans les 2 cas la balle va au gardien adverse.

Qu'il marque ou pas, la position de tous les joueurs est réinitialisée une fois qu'il a tiré (on parcourt les 2 listes et on diminue ou augmente de 5 la position tous les joueurs qui n'ont pas une position qui correspond à celles de base).

S' il clique sur P, il doit choisir à qui il va faire la passe. Il doit ensuite cliquer sur flèche de droite pour la faire au joueur à sa droite, flèche de gauche pour le joueur à sa gauche et espace pour le joueur devant lui. Si il est tout à droite ou tout à gauche il ne peut pas faire la passe à droite ou à gauche, on vérifie donc que dans la liste de son équipe il y'a un joueur à a sa position + 1 ou -1 en fonction de si c'est l'équipe 1 ou 2 ou si c'est à droite ou à gauche, avant de passer. Si il y'a un défenseur, un milieu ou personne de son équipe en face de lui, il ne peut pas faire la passe devant. On vérifie donc d'abord le poste du joueur (le gardien et le défenseur peuvent toujours faire la passe devant et l'attaquant jamais) puis pour le milieu on vérifie si l'un des attaquants est à seulement 5 de distance par rapport à lui avant de passer. S'il a dribblé son adversaire, il ne peut plus faire la passe à droite ou à gauche, mais étant donné que la position des autres joueurs ne changent pas, le test du -1 ou +1 permettra de gérer ce cas.

Si il clique sur D, donc le bouton dribbler dans la fonction input , cela lancera la méthode dribbler pour le joueur qui a la balle contre un joueur choisi en entrée. Pour vérifier, si il peut dribbler et qui il doit dribbler on vérifie d'abord son poste, seul les attaquants et les milieux peuvent dribbler. On vérifie ensuite qu'il y a un joueur en face de lui avant de le dribbler. Si c'est un attaquant qui a le ballon, on doit vérifier s' il y a un défenseur adverse, on doit donc parcourir la liste des défenseurs adverses et vérifier s' il y a un défenseur qui a la même position que l'attaquant au négatif + ou - 20 en fonction de l'équipe. Si c'est un milieu, on vérifie s' il ya un milieu adverse en face, donc on parcourt la liste des milieux et on vérifie s' il ya un milieu à la même position au négatif que le possesseur de la balle. Si toutes les conditions sont respectées il dribble l'adversaire, s' il réussit il doit faire l'action suivante, sinon la balle est donnée au gardien adverse et les positions sont réinitialisées.

Des textes permettant d'aider l'utilisateur s'affichent à l'écran en fonction de la touche qu'il clique (s'il tire alors qu'il ne peut pas ou essaie de passer alors qu'il ne peut pas… etc).

Pour représenter la balle graphiquement, on prend les positions de l'image de la balle correspondent à celles du possesseur de la balle.

Procédure d'installation (Linux)

Tout d'abord il faut installer les librairies SFML, il faut taper sur le terminal :

```
sudo apt-get install libsfml-dev
```

Et on a juste à taper make dans le terminal en se plaçant dans le dossier Projet_Draft (il faudra peut-être supprimer .depends avant de make).

Pour les tests unitaires commenter le premier main et décommenter le reste et #define CATCH_CONFIG_MAIN et #include "catch.hpp".

Contraintes

Nous avons 10 classes.
On utilise des vector et des listes dans la classe Jeu.
On utilise une surcharge d'opérateur (<<) pour afficher les joueurs et leurs détails dans le terminal.
Nous avons 3 fonctions virtuelles, une pour les postes des joueurs, une pour avoir leur vitesse et une toCSV.
Nous avons plus de 3 niveaux de hiérarchie (4).
Nous avons des tests unitaires.